Automatic Locomotion Pattern Generation for Modular Robots

Akiya Kamimura*, Haruhisa Kurokawa*, Eiichi Yoshida*
Kohji Tomita*, Satoshi Murata† and Shigeru Kokaji*

* National Institute of Advanced Industrial Science and Technology (AIST)
1-2-1 Namiki, Tsukuba, Ibaraki, 305-8564 Japan
{kamimura.a, kurokawa-h, e.yoshida, k.tomita, s.kokaji}@aist.go.jp

† Tokyo Institute of Technology
4259 Nagatsuta-cho, Midori-ku, Yokohama, 226-8502 Japan
murata@dis.titech.ac.jp

Abstract

Locomotion is considered as most basic function of robots. In the case of ordinary robots, they are not needed to change locomotion pattern because their configurations are constant. For self-reconfigurable modular robots, since they can change their configurations, locomotion patterns must be prepared in advance and changed by each configuration. There are two types of locomotion used for modular robots. One is locomotion by self-reconfiguration which is realized by using its reconfiguration capability. The other is locomotion using many degrees of freedoms of the configuration, e.g. walking, crawling and rolling, where the connection relationship between modules is constant. In this paper, we focus on the latter type of locomotion. Actually to design locomotion pattern suited for each configuration analytically or manually by human is difficult because it includes many DOFs. To solve this problem, we propose an automatic locomotion pattern generation method using neural oscillator and network and evolutionary computation method. The method is applicable for various kinds of modular robots. We confirmed the availability of the method by software simulation and hardware experiments.

1 Introduction

In recent years the feasibility of reconfigurable robotic systems has been examined through hardware and software experiments [1-14]. Self-reconfigurable robots (modular robots) proposed so far are composed of homogeneous or heterogeneous robotic modules and they can be connected together in a variety of configurations according to given tasks. They can also change their configuration by themselves by disconnecting connections between modules and changing positions of modules. This capability is effective for adapting themselves to the external environment by changing their configurations or repairing themselves by using spare modules. Modular robots seem to be useful at extreme conditions such as on distant planet, in deep sea, inside nuclear plants and at disaster areas where the access is difficult for human.

Current research topics on modular robots are mainly on their hardware systems and also many studies on their reconfiguration algorithms or planning methods have been proposed [15-20]. Locomotion using self-reconfiguration is actually useful when the number of modules becomes larger. However when the number of modules is not large, locomotion such as walking, crawling and rolling is faster and efficiently compared to the locomotion using reconfiguration. There are few studies on the latter locomotion using real modular robots or methods for making locomotion patterns in variety of module configurations. This is because locomotive motions need high motor torque for supporting the whole body or moving by themselves, which is difficult for most of the current modular robots. In [7], we have shown hardware experiments on various locomotive motions and reconfiguration between configurations by using our M-TRAN1 module, where all the sequences in each configuration were programmed by human and it needed much time and effort to make stable locomotion patterns.

On the other hand, in biological cybernetics research field, there are several researches on generation of biped or quadruped locomotion by using neural oscillators [21-24]. The method has been studied for understanding of the mechanism for walking from the neurodynamics point of view or realizing a robust locomotion and adaptation against the external disturbances. It is considered possible to apply the same principle on the modular robots for making locomotive motions.

In this paper, we describe an automatic locomotion generation method (called ALPG hereafter) aimed at making locomotion of arbitrary module configurations using neural oscillator as a model of CPG (Central Pattern Generator) and Genetic Algorithm for evolving parameters. In section 2 basic functions of M-TRAN module are explained; in section 3 the details of ALPG software are described and the results are shown and in section 4 hardware experiments on locomotion are described.

2 Basic Functions of M-TRAN Module

In this paper, we work with the self-reconfigurable modular robot M-TRAN2 shown in Fig.1 as an exercise for realizing locomotion of modular robot. This module is composed of three components, two semi-cylindrical parts and a link part. Each semi-cylindrical part can rotate from –90 to 90 degree independently by a geared motor embedded in the link. There are four permanent magnets on each of three connecting surfaces of each semi-cylindrical part. As the polarity of the magnets between two parts is different, the module can connect to other modules by magnetic force. As each

connecting surface can be connected to another connecting surface in every orthogonal relation, various lattice structures are easily formed as shown in Fig.2 and it can be reconfigured by changing positions of the semi-cylindrical parts.

Besides self-reconfiguration, this modular robot system can also make various robotic motions such as a crawler and a quadruped robot [7] by using two degrees of freedoms on each module.

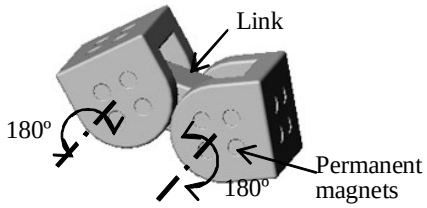

Figure 1. Schematic view of the module

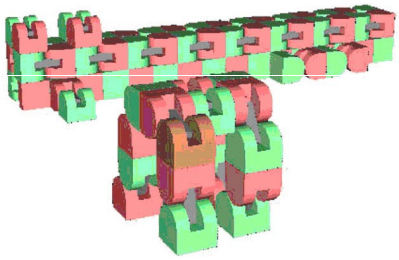

Figure 2. Example of possible configurations

3 Automatic Locomotion pattern Generation Method

3.1 Locomotion Generation Flow

The automatic locomotion pattern generation method (ALPG) makes locomotion pattern of an arbitrary module configuration and an initial shape in simulation space. Note that the configuration here means connection relationship between modules and that shape will change according to each module's angle.

Figure 3 shows a flow chart for making locomotion pattern in ALPG software. We adopted the Vortex simulator (Critical Math Labs) as a three-dimensional dynamic simulator, which is fast enough to calculate dynamic motions in real time. First, a module configuration and an initial shape are determined. The locomotion pattern made by the ALPG software depends on the initial shape. In 3.4 we show an example of different locomotion patterns emerged from the same configuration but different initial shape. Second, the configuration is input to the software and each module's semi-cylindrical part begins to rotate periodically depending on frequency and amplitude determined by a neural oscillator. The locomotion pattern made by the neural oscillators is evaluated and GA (Genetic Algorithm) embedded in the ALPG software evolves the parameters for the oscillators.

The evolved locomotion pattern consists of motion sequences at each motor of each module, and the locomotion is realized in the real world by downloading these sequences into the real hardware.

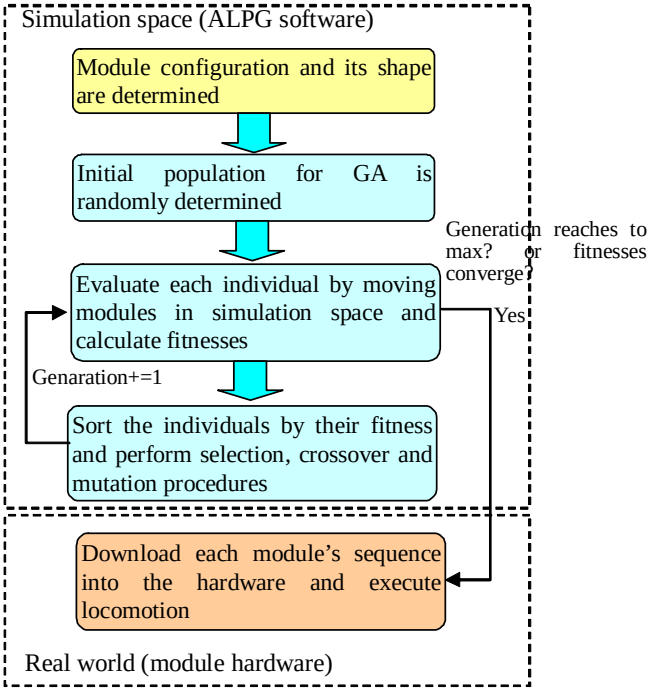

Figure 3. Simulation flow in ALPG software

3.2 Module and Environment Description in Simulation

In simulation space, the coordinate system shown in Fig.4 is used. The gravity goes toward the minus direction of the y-axis. The ground is completely flat and having appropriate friction. The rotation of each semi-cylindrical part is controlled by an output from the neural oscillator (described in 3.3). We carefully examined the maximum torque of the motor and relationship between the maximum rotation speed and the input value by using the real hardware and implemented them in the simulator. In simulation, connections between modules are also considered and two surfaces are automatically connected when they are drawing near. The simulator calculates collisions and friction between modules or a module and the ground, and the scene is updated in every 0.015sec (*step* hereafter) which is the calculation frequency of the simulation.

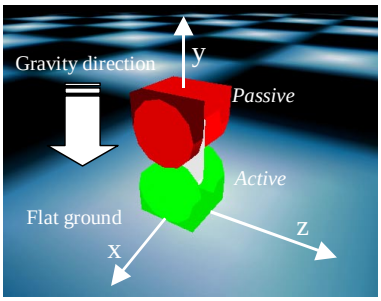

Figure 4. Simulation space

3.3 Neural Oscillator Model (CPG)

To realize stable locomotion where the rotation angles of the modules are cooperatively oscillated, we applied a neural oscillator as a model of the CPG (Central Pattern Generator) to control each module's rotation. Each neuron in this model is represented by the following non-linear differential equations (1), which is the same model used by Taga[22] and Kimura[23]. As shown in Fig.5, a couple of inhibiting neurons (CPG) is connected to each rotation motor of each module, which controls rotation in proportion to the output from a CPG expressed by equation (2). CPGs are mutually entrained and oscillate in the same period and with a fixed phase difference. This mutual entrainment between CPGs, other entrainment called global entrainment between CPGs and a mechanical system results in a cooperative motion with modules.

$$\begin{aligned} \tau \dot{u}_{\{1,2\}i} &= -u_{\{1,2\}i} - w_{12} y_{\{2,1\}i} - \beta v_{\{1,2\}i} + u_e + f_{\{1,2\}i} + s_{\{1,2\}i} \\ \tau' \dot{v}_{\{1,2\}i} &= -v_{\{1,2\}i} + y_{\{1,2\}i} \end{aligned} \quad (1)$$

$$\begin{aligned} y_{\{1,2\}i} &= \max(0, u_{\{1,2\}i}) \\ f_{1i} &= k(angle[step]_i - initialangle_i) \\ f_{2i} &= -f_{1i} \\ s_{\{1,2\}j} &= \sum_{j=1}^{n} weight_{ij} y_{\{1,2\}j} \end{aligned}$$

$$Output_i = -p_1 y_{1i} + p_2 y_{2i} \quad (2)$$

where u_i is the inner state of the ith neuron; v_i is a variable representing the degree of the self-inhibition effect of the ith neuron; y_i is the output of the ith neuron; u_e is an external input with a constant rate; f_i is a feedback signal from each angle. τ and τ' are time constants of u_i and v_i; $weight_{ij}$ is a connecting weight between the ith and jth neurons.

The differential equations above are solved by using Runge-Kutta method in every step, and every module's *angle*[*step*] at each step is stored as motion sequences and utilized for making locomotion on the hardware.

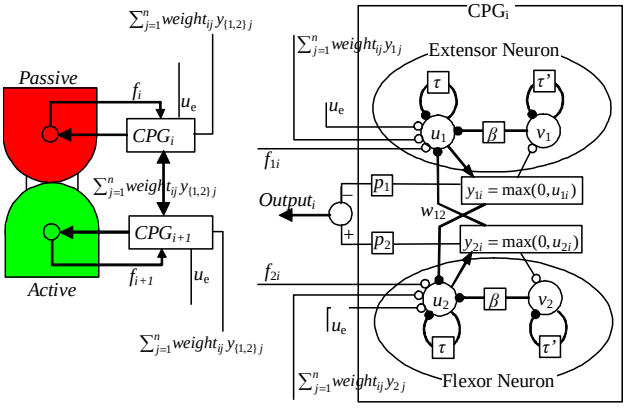

Figure 5. Schematics of the neural oscillator(CPG)

3.4 Evolutionary Computation

We implemented GA on the ALPG software to evolve the locomotion pattern automatically. By using GA, initial values $u_{0\{1,2\}i}$ and $v_{0\{1,2\}i}$ of each CPG and the connection weights $weight_{ij}$ are evolved together. The $u_{0\{1,2\}i}$ and $v_{0\{1,2\}i}$ are a real number from –8.0 to 8.0 and from 0.0 to 3.0 respectively. The $weight_{ij}$ is selected from three values, –1:inhibitory connection, 0: no connection and 1:excitatory connection. The initial values $u_{0\{1,2\}i}$ and $v_{0\{1,2\}i}$ are important parameters for converging the oscillation of the CPG to a limit cycle attractor smoothly. The connection weights determine the phase-contrast between neurons and make the limit cycle robust against external disturbance.

First, a group of parameters, $u_{0\{1,2\}i}$, $v_{0\{1,2\}i}$ and $weight_{ij}$, is randomly initialized by the number of population size, *pop_size*. A locomotion made by each individual is evaluated one by one in 15 sec in simulation space by the evaluation function represented by equation (3).

$$\begin{aligned} fitness = {} & 200 \cdot length - 250 \cdot width - 0.47 \cdot loss / num \\ & + 50 \cdot speed - 300 \cdot connection - 300 \cdot tumbled \end{aligned} \quad (3)$$

where *length* is a moving distance of the center of gravity in plus direction of z-axis; *width* is a moving distance of the center of gravity in x-axis; *loss* is a energy loss that is an accumulated value of motor torque during evaluation interval and *num* is the number of modules. The *speed* is an average speed in evaluation time. The value of *connection* increases when two modules' surfaces are connected, which inhibits the change of configuration. The value of *tumbled* becomes 1 when the acceleration of the center of gravity becomes larger, which inhibits tumble or unreasonable motions. By using above evaluation function, a locomotion pattern such that the module configuration moves faster along the z-axis in the positive direction with low energy consumption will emerge.

When every individual is evaluated, they are sorted by their *fitness* and the lower groups are deleted according to the selection rate, *s_rate*. To fill the deleted parts, crossover is achieved by selecting parents from the remaining individuals by a roulette selection method. As for initial values, UNDX (Unimodal Normal Distribution Crossover) method [25] is used. This method is used for real-coded Genetic Algorithm, which is superior in optimization for multimodal functions or variables having many local minimums. On the other hand, N-point crossover method is used as for connection weights. In mutation procedure, several individuals are selected according to the mutation rate, *m_rate*, and initial values, $u_{0\{1,2\}i}$ and $v_{0\{1,2\}i}$, of each individual are given a little bit fluctuation and a part of the connection weights is randomly initialized by -1, 0 or 1. Hereafter the procedure restarts with the newly generation. The process of GA stops when the number of generation has passed a maximum number of generation *max_gene* or the average of *fitness* becomes constant. By repeating above GA processes, quasi-optimized locomotion pattern will emerge. The fixed parameters of the neural oscillator

and GA are summarized in Table 1.

Table 1. Parameters for neural oscillator (N.O.) and GA

Parameters for N.O.	Value	Parameters for GA	Value
τ	0.05	*pop_size*	150
τ'	0.6	*max_gene*	150
β	1.5	*s_rate*	0.6
p_1, p_2	0.125	*m_rate*	0.05
k	8		
w_{12}	2.5		
u_e	8.5		

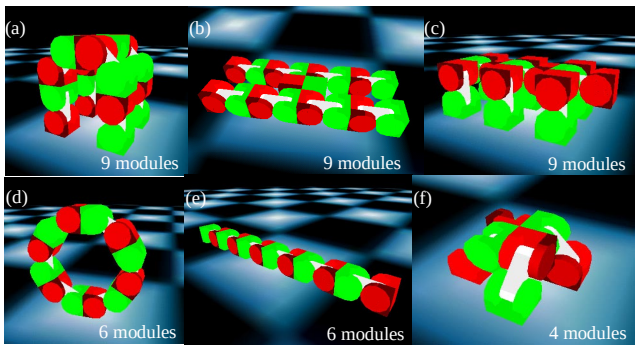

Figure 6. Examples of tested configurations

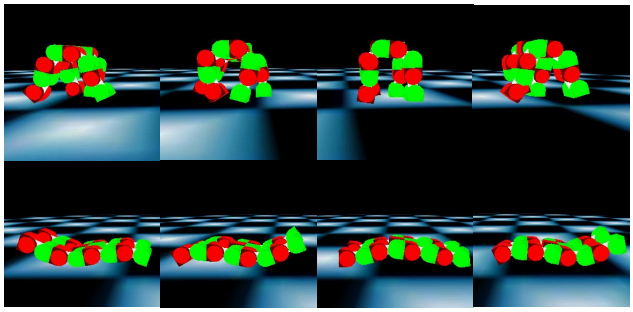

Figure 7. Obtained locomotion patterns, gait pattern (upper) and wave-like pattern (below)

3.5 Application to Various Module Configurations

We applied ALPG method to various module configurations shown in Fig.6. For every configuration stable locomotion patterns were obtained. The obtained locomotion patterns of Fig.6 (a) and (b) are shown in Fig.7. In spite of the configurations being the same, two different kinds of locomotion patterns were obtained, one is a gait pattern and the other is a wave-like motion. This is caused by the difference of the initial shape.

Figure 8 shows the fitness curve of each configuration from Fig.6 (a) to (f). It is found that the configuration (d) has the highest fitness value. On the flat grounds, it is considered that the crawler shape is more effective for moving faster since it can roll to move.

Figure 9 (a) shows the relationship between angle and angular velocity for one of the motors involved in Fig.6 (a). Figure 9 (b) shows the transition of every motor's angle of Fig.6 (a). It is found that every motors is oscillating with a constant frequency (about 1.2Hz), amplitude and phase-contrast, namely the locomotion pattern is stable. The needed time for making locomotion pattern by ALPG software depends on the number of modules and the number of collisions at each step. It took about 6 hours by using 2.53GHz Pentium 4 processor PC to evolve a stable walking pattern for the 9-module configuration in Fig.6 (a).

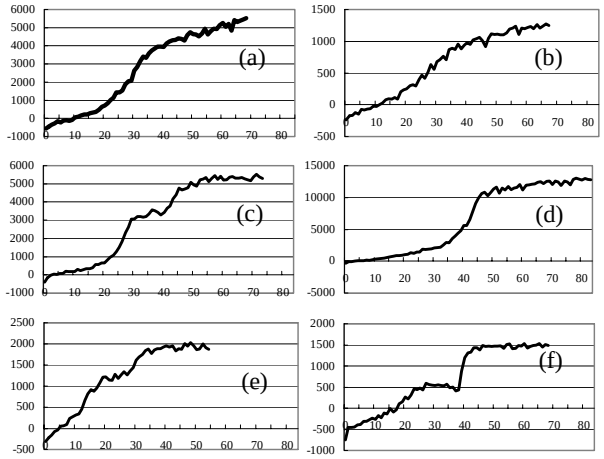

Figure 8. Fitness vs. generation of various configurations shown in Fig.6

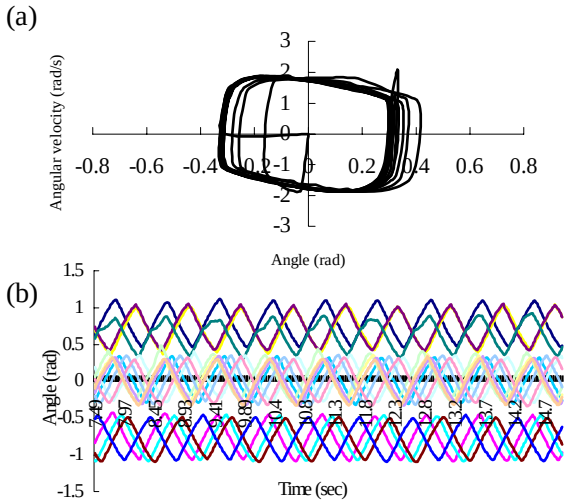

Figure 9. (a) Relationship between angle and angular velocity for one of the motors involved in Fig.6 (a). (b) Transition of every motor's angle in Fig.6 (a).

4 Hardware Experiments

4.1 Specifications of M-TRAN2 Module

We have developed twenty M-TRAN2 modules shown in Fig.10. There are two semi-cylindrical parts called a passive part and an active part. The passive part has four permanent magnets (S pole outside) on each of three surfaces. On the same surface, electrodes are placed symmetrically for power supply (VCC, GND), global communication (RS-485) and local communication. Inside the passive part, there are circuit boards including a microprocessor

(Neuron chip, TMPN3120FE5M, Echelon Corporation) for global inter-module communication and a microprocessor (PIC16F873, Microchip Technology, Inc.) for local communication. A power supply circuit and a battery are also embedded as shown in Fig.11. Power for the module is supplied by an internal battery or by connecting wires from outside to any of surfaces of the modules. In the experiments that follows we used the internal battery and no tethers were attached.

Inside the active part, there are connecting plates that will rise to the surface by the attractive force of the magnets and connects two surfaces electrically and mechanically. Two surfaces are also detached automatically by heating and lengthening shape memory alloy coils by small light bulbs as shown in Fig.11. There is also a microprocessor (PIC16F873) for controlling detachment and local communication with other modules.

Inside the link, there are two geared motors and their control circuit board that includes a microprocessor (PIC16F877) into which we implemented a PID position control program. The motion sequence of each module made by the ALPG software is realized by a trajectory control using this PID position control.

Specifications of M-TRAN2 module are summarized in Table 2. More details on mechanical and electrical design of M-TRAN2 module are available in [26].

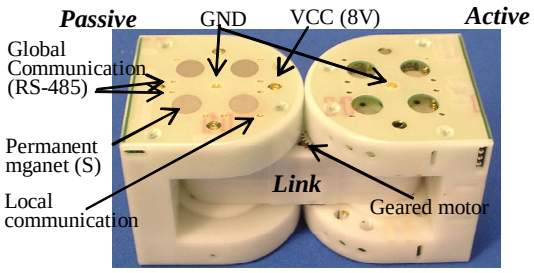

Figure 10. M-TRAN2 module

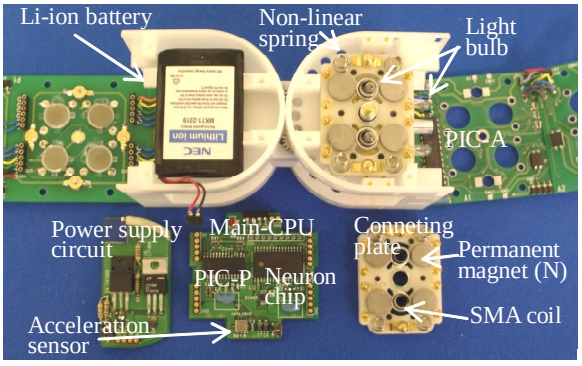

Figure 11. Inner structure of the module

4.2 Experimental Setup

Figure 12 shows the experimental setup and the internal communication system of the modules. First, the motion sequence of each module made by the ALPG software is downloaded by using global communication line between host PC and modules and it is stored in Main-CPU RAM of each module. After the download is completed, synchronization between modules is achieved by host PC and the cable is disconnected. In every 60 msec each module's Main-CPU sends the angle datum for each step to the microprocessor (PIC-M) in link and the locomotion of the modules is thereby realized in a cooperative manner.

Table 2. Specifications of a M-TRAN2 module

Item	Value
Dimension	60x120x60mm
Weight	0.4kg (including battery)
CPU	Neuron chip and three PICs
Global communication	LonWorks, 39kbps
Local communication	4,800 bps
Power supply (wired)	DC 8V~20V
Power supply (battery)	DC 3.8V
Max. torque of each axis	19.8 kg cm (rating)
Max. rotation speed	0.5π rad/sec
Connecting force	83 N
Battery	Li-ion (3.8V, 700mAh)
Total power dissipation	0.4W(8V)
Sensor	Acceleration sensor (3 axes)

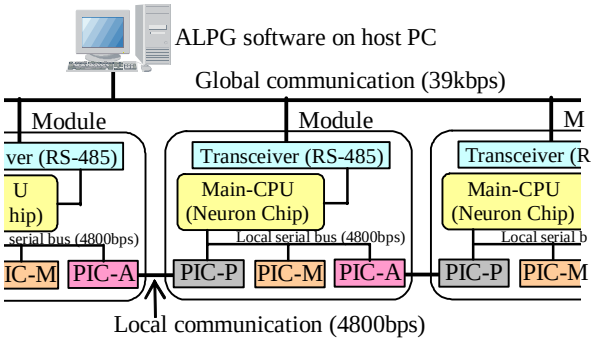

Figure 12. Experimental setup and communication system

4.3 Experiments on Locomotion

We have performed experiments on locomotion of all the configurations shown in Fig.6. As shown in Fig.13, all the locomotion but Fig.6 (d) are successfully realized by real hardware. The moving speeds of locomotion in hardware experiments and in simulations are shown in Table 3. This proves the validity of the simulation and the implemented model. As for the configuration of Fig.6 (d), the chain was broken by a module detachment and we stopped the experiment. One of the reasons for the failure is that synchronization between modules was not

Table 3. Comparison of the speed in hardware experiments and in simulation

	Hardware experiments	Simulation
(a)	20 cm/sec	23 cm/sec
(b)	4.5 cm/sec	6.8 cm/sec
(c)	11cm/sec	19 cm/sec
(d)	N/A	49.3 cm/sec
(e)	6.0 cm/sec	7.9 cm/sec
(f)	6.0 cm/sec	6.7 cm/sec

complete. It seems to be solved by doing synchronization at each step by using the global communication between modules.

5 Concluding Remarks and Future Works

In this paper, we applied neural oscillator as a CPG model for making stable locomotion patterns of modular robotic systems. By using Genetic Algorithm we have succeeded in making a locomotion pattern suited for a given module configuration and its shape automatically. As it is also possible to make the locomotion patterns such as going to left, right and back by changing the evaluation function, we will be able to control the moving direction of the module configuration. Furthermore, we performed hardware experiments on various locomotion patterns by implementing the obtained patterns in real hardware and realized locomotion successfully. We are convinced that this method is also applicable to other modular robotic systems by changing the module description in the simulator. In the current M-TRAN2 system, self-reconfiguration capability has not been used. Our next target is to realize locomotion utilizing both self-reconfiguration capability and motion capability of M-TRAN2 system. As described in section 1, locomotion using self-reconfiguration costs too much in time and energy consumption for current modular robotic systems. However that locomotion becomes more effective compared to walking or rolling when the number of modules increases. As future works, we'd like to develop a modular robotic system that can move around with a small size of configuration for searching or any other tasks and can change into a large size of configuration for crossing a wall or a ditch by connecting with each other if necessary.

Acknowledgement

This study has been supported by Science and Technology

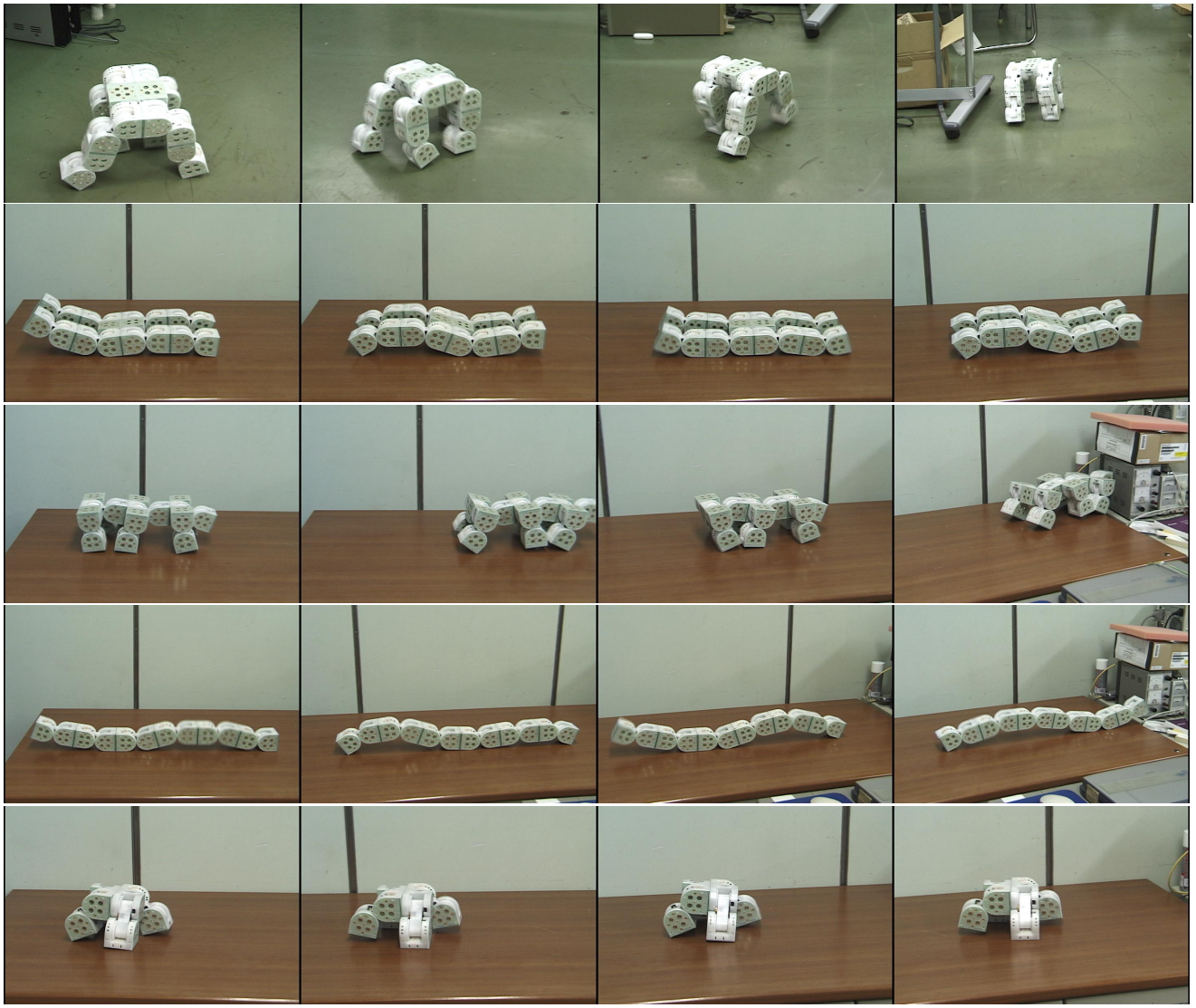

Figure 13. Hardware experiments on various locomotion patterns
The sequence of each module is downloaded in advance and in the experiments all the modules are moving by internal battery and no tethers are attached.

Research Grant Program for Young Researchers with a Term from Ministry of Education, Culture, Sports, Science and Technology (MEXT) of Japan.

As for neural oscillator (CPG) and its implementation we have been advised by Dr. Sooyol Ok at Communications Research Laboratory, Information and Network Systems Division, Keihanna Human Info-Communications Research Center, Image Group, Japan.

References

[1] Y.Kawauchi, M.Inaba and T.Fukuda, A study on cellular robotic system (A realization of robotic system capable of adaption, self-organization, and self-evolution), RSJ, vol.12, 116/132 (1994), in Japanese.

[2] G.S.Chirikjian, et al., Evaluating Efficiency of Self-Reconfiguration in a Class of Modular Robots, J.Robotic Systems, 12-5, 317/338 (1995).

[3] S.Murata, et al., Self-assembling machine, Proc. IEEE ICRA, 441/448 (1994).

[4] S.Murata, et al., A 3-D self-reconfigurable structure, Proc. IEEE ICRA, 432/439 (1998).

[5] E.Yoshida, et al., Micro self-reconfigurable robotic system using shape memory alloy, DARS2000, 145/154 (2000).

[6] S.Murata, et al., Hardware Design of Modular Robotic System, Proc. IEEE IROS, 2210/2217 (2000).

[7] A.Kamimura, et al., Self-reconfigurable Modular Robot – Experiment on reconfiguration and locomotion, Proc. 2001 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS2001), 606–612, 2001.

[8] K.Hosokawa, et al, Self-Organizing Collective Robots with Morphogenesis in a Vertical Place, Proc. IEEE ICRA, 2858/2863 (1998).

[9] D.Rus and M.Vona, A basis for self-reconfigurable robots using crystal modules, Proc. IEEE IROS, 2194/2202 (2000).

[10] K.Kotay, et al., The self-reconfigurable robotic molecule, Proc. IEEE ICRA, 424/431 (1998).

[11] C.Ünsal, H.Kiliccote and K.Kohsla, I(CES)-cubes; a modular self-reconfigurable bipartite robotic system, Proc. SPIE, vol.3839, 258/269 (1999).

[12] A.Castano and P.Will, Mechanical design of a module for reconfigurable robots, Proc. IEEE IROS, 2203/2209 (2000).

[13] M.Yim, New Locomotion Gaits, Proc. IEEE ICRA, 2508/2514 (1994).

[14] A.Casal and M.Yim, Self-reconfigurable planning for a class of modular robot, Proc. SPIE, vol.3839, 246/257 (1999).

[15] E.Yoshida, et al., A distributed method for reconfiguration of 3-D homogeneous structure, Advanced Robotics, 13-4, 363/380 (1999).

[16] K.Tomita, et al., Self-assembly and self-repair method for distributed mechanical system, IEEE Trans. on Robotics and Automation, 15-6, 1035/1045 (1999).

[17] K.Kotay and D.Rus, Motion synthesis for the self-reconfigurable molecule, Proc.1998 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 843/851 (1998).

[18] C.Ünsal, et al., A modular self-reconfigurable bipartite robotic system: implementation and motion planning, *Autonomous Robots*, 10-1, 23/40 (2001).

[19] Z.Butler, et al., Generic Decentralized Control for a Class of Self-Recon¯gurable Robots, Proc. IEEE ICRA, 809/816 (2002).

[20] K.C.Prevas, et al., A Hierarchical Motion Planning Strategy for a Uniform Self-Reconfigurable Modular Robotic System, Proc. IEEE ICRA, 787/792 (2002).

[21] K.Matsuoka, Mechanisms of frequency and pattern control in the neural rhythm generators, Biolog. Cybern., 56, 345/353 (1987).

[22] G.Taga, A model of the neuro-musculo-skeletal system for human locomotion II – real-time adaptability under various constraints, Biolog. Cybern., 73, 113/121 (1995).

[23] H.Kimura, et al., Realization of dynamic walking and running of the quadruped using neural oscillator, Autonomous Robots, 7-3, 247/258 (1999).

[24] K.Hase, et al., Development of three-dimensional whole-body musculoskeletal model for various motion analyses, JSME Int J C 40, 25/32 (1997).

[25] I.Ono and S. Kobayashi, A Real-coded Genetic Algorithm for Function Optimization Using Unimodal Normal Distribution Crossover, Proc. 7th ICGA, 246/253 (1997).

[26] H.Kurokawa, et al., Self-Reconfigurable Modular Robot (M-TRAN) and its Motion Design, Proc. ICARCV 2002 (will appear).